МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студент гр. 8304 ____ Нам Ё Себ

Преподаватель ____ Фирсов М.А.

Санкт-Петербург

2019

Цель работы.

Изучить основы рекурсии и составления эффективных алгоритмов.

Постановка задачи.

1) Разработать программу, использующую рекурсию;
2) Сопоставить рекурсивное и итеративное решение задачи;
3) Сделать вывод о целесообразности и эффективности рекурсивного подхода для решения данной задачи.

Вариант 16

Построение синтаксического анализатора для понятия *скобки*.

скобки::= A | B | (*скобки скобки*)

Описание алгоритма.

Для решения поставленной задачи была реализована рекурсивная функция check, которая заменяет каждую подстроку “(x x)” полученной строки на подстроку “x” и рекурсивно вызывает саму себя от измененной строки. Выход из рекурсии осуществляется посредством двух проверок. Первая – проверка на то, что полученная строка равна строке “x” (так как дальнейшая замена бесполезна) и возвращает значение True, вторая – проверка на подстроки (x x) в полученной строке, если на одном из шагов таковой не было найдено, то функция возвращает значение False.

Спецификация программы.

Программа предназначена для синтаксического анализа выражения методом рекурсии.

Программа написана на языке C++. Входными данными является либо строка, либо путь до файла содержащего строку (строки). Выходными данными являются промежуточные значения входной после замены описанной выше и глубина рекурсии.

Описание функций.

1) Функция CheckInputData.

Объявление функции:

```
bool CheckInputData(std::string inputData);
```

Данная функция осуществляет проверку на отсутствие нешаблонных символов во входной строке. Если такие имеются возвращаемое значение – False, иначе – True. Обход по принятой строке осуществляется при помощи цикла for, проверка на нешаблонные символы осуществляется при помощи условного оператора if.

2) Функция Analyzer.

Объявление функции:

```
bool Analyzer(std::string& _data_, std::ostream& out, int i)
```

Данная функция осуществляет проверку на то, удовлетворяет ли входная строка поставленной задаче. Функция принимает проверяемую строку, ссылку на поток вывода и имеет параметр i, который по умолчанию равен 0, он используется для подсчета глубины рекурсии. Сначала функция проверяет входную строку на равенство строке “x”, если это условие выполнилось, то функция завершается, возвращая True. Далее при помощи метода find класса std::string осуществляется поиск подстроки “(x x) ” (возвращаемое значение сохраняется в переменную obj), если данная подстрока не была найдена (то есть функция вернула специальный параметр std::string::npos), то функция завершается, возвращая False. Если последняя проверка не выполнилась, то подстрока “(x x)” была найдена и индекс ее первого вхождения в строку workData содержится в переменной obj. Далее при помощи метода erase класса std::string из строки workData удаляется найденная ранее подстрока “(x x)”. Далее при помощи метода insert класса std::string осуществляется “вставка” подстроки “x” в строку workData (места вставки и удаления определяются при помощи ранее найденного индекса). В конце функция рекурсивно вызывает саму себя от уже измененной строки.

Вывод.

Был получен опыт работы с рекурсией и с построением синтаксического анализатора. На мой взгляд, итеративное решение поставленной задачи более эффективно.

ПРИЛОЖЕНИЕ

1) ТЕСТИРОВАНИЕ:

Работа программы для строки (A ((A B) (A A))):

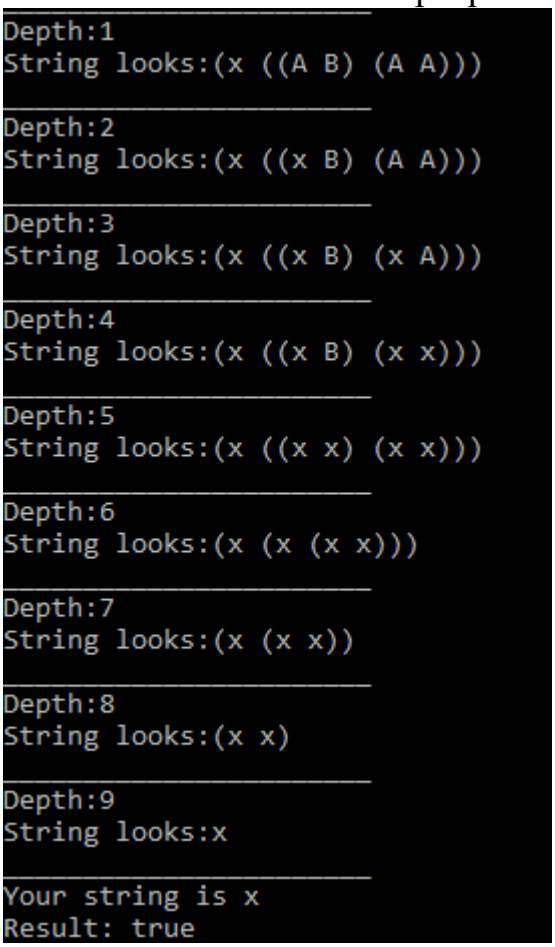

Таблица результатов ввода/вывода тестирования программы:

Ввод	Вывод
(A B)	true
((A C))	false
))A B((	false
(A (B (A (A (A B)))))	true
(A A (B B (A A (B A))))	false
A (B A) B	false
(A B (C A) W))	false
(A ((A B) (A A)))	true
((B ((B A) (B B))) (A ((A B) (A A))))	true
(B A (A B) (A B) A B ((A B) A B) A B) A B)	false

2) ИСХОДНЫЙ КОД:

```
#include <iostream>
#include <string>
#include <fstream>

std::string longLine = "_____________________________________________________________________________";
std::string shortLine = "_____________________";
std::string enterDialog = "0";
std::string fromConsole = "1";
std::string fromFile = "2";
std::string exit_or_back = "3";
std::string finishProgramm = "100";

/* Функция для ведения диалога с пользователем по поводу формата вывода
(Сделана для того, чтобы в случае ошибки пользователь мог ввести данные еще раз); */
std::string outDialog(std::string option, int flag);

/* Функция для ведения диалога с пользователем по поводу формата ввода
(Сделана для того, чтобы в случае ошибки пользователь мог ввести данные еще раз); */
std::string inDialog(std::string option, int flag);

// Функция, ведущая общий диалог с пользователем и объединяющая данные из outDialog и inDialog;
std::string dialog(std::string inOption, int flag);

// Функция, проверяющая введеную строку на соотвествующие символы
bool checkInputData(std::string inputData);

// Функция, выводящая результат и запускающая функцию analyzer;
void output(std::string workData, std::ostream& out, int i);

// Рекурсивная функция, обрабатывающая введеную пользователем строку;
bool analyzer(std::string& workData, std::ostream& out, int i);

std::string outDialog(std::string option, int flag) {
	std::cout << "\nChoise your output format(Press a number and after press Enter) :\n\n";
	std::cout << "1)Console\n";
	std::cout << "2)Your file(Write a path for your file or write name of file and it will be
created in a directory with a code)\n";
	std::cout << "3)Back\n";
	std::cout << "Your choise: ";
	std::string inputData;
	if(flag == 1)
		option = fromFile;
	else
		std::cin >> option;
	if (option == fromConsole)
	{
		return "cout";
	}
	else if (option == fromFile)
	{
		std::cout << "\nInput path for your file.txt and press Enter: ";

		if(flag == 1)
			inputData = "./Tests/TestResult.txt";
		else
		{
			std::cin.ignore();
			std::getline(std::cin, inputData);
```

```
                size_t pos = inputData.find(".txt");
                if (pos != std::string::npos)
                        inputData.append(".txt");
        }
        std::ofstream outFile;
        outFile.open(inputData);
        if (!(outFile.is_open()))
        {
                std::cout << "Incorrect path! Please, try again!\n";
                std::cout << std::endl << longLine;
                return outDialog(option, flag);
        }
        return inputData;
    }
    else if (option == exit_or_back)
    {
            std::cout << std::endl << longLine << std::endl;
            return dialog(enterDialog, flag);
    }
    else
    {
            std::cout << longLine << std::endl  << "Please press not like a dunmb, only three
numbers're here -_-" << std::endl << longLine ;
            return outDialog(enterDialog, flag);
    }
}

std::string inDialog(std::string option, int flag)
{
    std::string inputData;
    if (option == fromConsole)
    {
            std::cout << "\nInput your string and press Enter: ";
            std::getline(std::cin, inputData);
            std::cout << longLine;
            if (!checkInputData(inputData))
            {
                    std::cout << "\nIncorrect symbols, please, try again!\n";
                    std::cout << std::endl << longLine;
                    return inDialog(option, flag);
            }
            else
                    return inputData;
    }
    else if (option == fromFile)
    {
            std::cout << "\nInput path for your file.txt and press Enter:";
            if(flag == 1)
                    inputData = "./Tests/TestFile.txt";
            else
                    std::getline(std::cin, inputData);
            std::ifstream inputFile(inputData);
            if (!(inputFile.is_open()))
            {
                    std::cout << "Incorrect path! Please, try again!\n";
                    std::cout << std::endl << longLine;
                    return inDialog(option,flag);
            }

            else
            {
                    std::cout << std::endl << longLine << std::endl  << "IT WORKS!" << std::endl <<
longLine;
```

```
                        return inputData;
                }
        }
}

std::string dialog(std::string inOption, int flag)
{
        std::cout << "Choise your input format(Press a number and after press Enter):\n\n";
        std::cout << "1)Console\n";
        std::cout << "2)Open your file(Write the path to your file)\n";
        std::cout << "3)Exit\n";
        std::cout << "Your choise: ";

        std::string outOption = "0";
        std::ofstream out;
        std::string inputData;
        std::string outPath;

        if(flag == 1)
                inOption = fromFile;
        else
        {
                std::cin >> inOption;
                // Чтобы, функция std::getline() работала корректно;
                std::cin.ignore();
                std::cout << longLine;
        }


        if (inOption == fromConsole)
        {
                inputData = inDialog(fromConsole, flag);
                outPath = outDialog(outOption, flag);
                if (outPath == "cout")
                {
                        std::ostream& workStream = std::cout;
                        output(inputData, workStream, 0);
                }
                else
                {
                        out.open(outPath);
                        std::ostream& workStream = out;
                        output(inputData, workStream, 0);
                }
        }
        else if (inOption == fromFile)
        {
                inputData = inDialog(inOption, flag);
                std::string dataFile;
                std::ifstream inputFile(inputData);
                outPath = outDialog(outOption, flag);
                if (outPath == "cout")
                {
                        std::ostream& workStream = std::cout;
                        while (std::getline(inputFile, dataFile))
                        {
                                if (!checkInputData(dataFile))
                                {
                                        workStream << std::endl << longLine << std::endl << "Your input: " +
dataFile << std::endl << "Result: False - incorrect symbols\n" << std::endl << longLine;
                                        continue;
                                }
                                output(dataFile, workStream, 0);
```

```
                        }
                }
                else
                {
                        out.open(outPath);
                        std::ostream& workStream = out;
                        while (std::getline(inputFile, dataFile))
                        {
                                if (!checkInputData(dataFile))
                                {
                                        workStream << std::endl << longLine << std::endl << "Your input: " +
dataFile << std::endl << "Result: False - incorrect symbols\n" << std::endl << longLine;
                                        continue;
                                }
                                output(dataFile, workStream, 0);
                        }
                }

        }
        else if (inOption == exit_or_back)
        {
                return finishProgramm;
        }
        // В случае неккоректного выбора параметра(опции);
        else
        {
                std::cout << std::endl << "\nPlease press not like a dunmb, only three numbers're here -
_-\n"  << std::endl << longLine << std::endl;
                return dialog(enterDialog, flag);
        }

        char quest;
        std::cout << std::endl << "Mission complete!" << std::endl;
        std::cout << std::endl << "Do you want to retry?(y/n)" << std::endl;
        std::cin >> quest;
        std::cout << longLine << std::endl << std::endl;
        if (quest == 'y')
                return dialog(enterDialog, flag);
        else
                return finishProgramm;
}

bool checkInputData(std::string inputData)
{
        for (int i = 0; i < inputData.size(); i++)
                if (inputData[i] != 'A' && inputData[i] != 'B' && inputData[i] != '(' && inputData[i] !=
')' && inputData[i] != ' ' && inputData[i] != '\0')
                        return false;
        return true;
}

void output(std::string workData, std::ostream& out, int i)
{
        out << std::endl << longLine << std::endl;
        out << std::endl << "Your string is " + workData + "\nResult: " << std::boolalpha <<
analyzer(workData, out, i);
        out << std::endl << longLine << std::endl;
}

bool analyzer(std::string& workData, std::ostream& out, int i)
{
        out << std::endl << "Depth:" << i << std::endl << "String looks:" << workData << std::endl <<
shortLine;
```

```
	if (workData == "x")
		return true;
	std::size_t objA = workData.find("A");
	std::size_t objB = workData.find("B");
	if (objA != std::string::npos)
	{
		workData.erase(objA + workData.begin(), objA + workData.begin() + 1);
		workData.insert(objA + workData.begin(), 'x');
		return analyzer(workData, out, ++i);
	}
	else if (objB != std::string::npos)
	{
		workData.erase(objB + workData.begin(), objB + workData.begin() + 1);
		workData.insert(objB + workData.begin(), 'x');
		return analyzer(workData, out, ++i);
	}
	else
	{
		size_t obj = workData.find("(x x)");
		if (obj == std::string::npos)
			return 0;
		workData.erase(obj + workData.begin(), obj + workData.begin() + 5);
		workData.insert(obj + workData.begin(), 'x');
		return analyzer(workData, out, ++i);
	}

}

int main(int argc, char* argv[])
{
	std::cout << "\tHELLO! THIS IS SUPER SYNTAX ANALYZER!" << "THIS IS PROGRAMM FOR ANALYZE YOUR
STRING IF IT IS x::= A|B|(xx)\n\n";
	int flag = 0;
	if(argc == 2)
		flag = 1;

	if (dialog(enterDialog, flag) == finishProgramm)
		std::cout << "Goodbye, CYA later!";
	else
		std::cout << "Impooooosible, errrooooorrrrr";

	return 0;
  }
```